Microprocessor

Third year Comm.

Lect.4

Dr. Roayat Ismail

Objectives

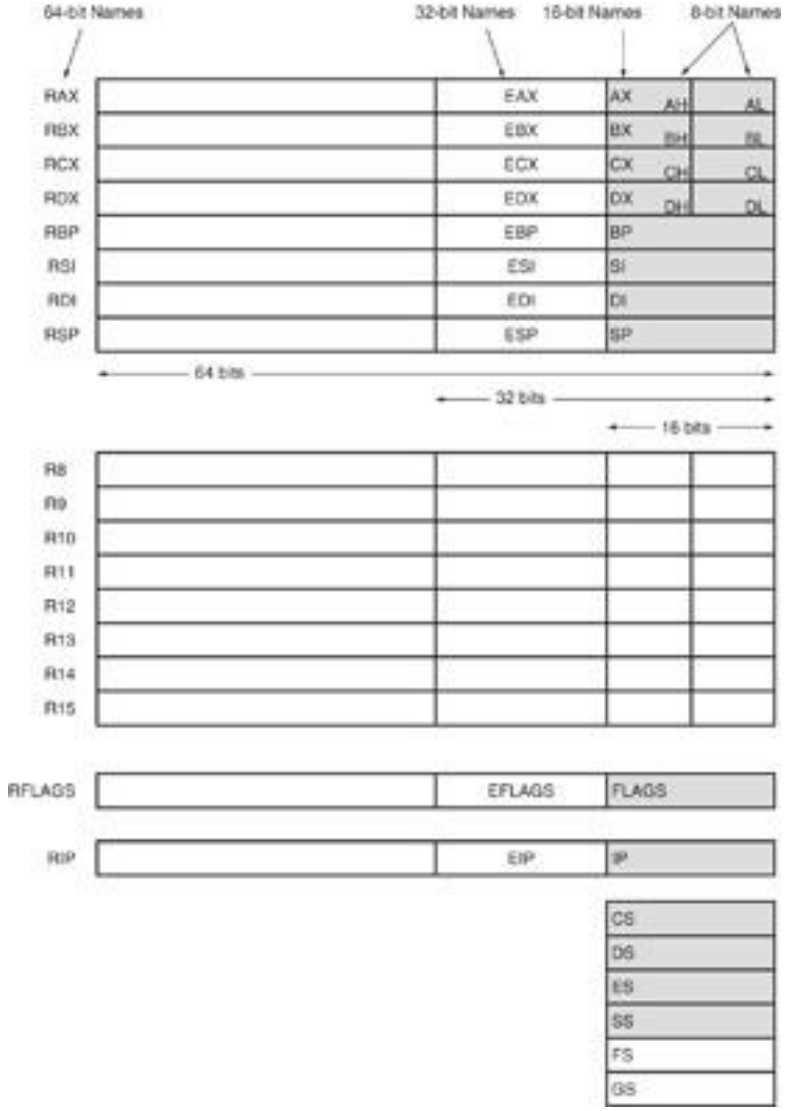
- Revise function and purpose of each program-visible register in the 8086 microprocessors, including 64-bit extensions.
- Describe how memory is accessed using protected mode memory-addressing techniques.
- Describe program-invisible registers
- memory-paging mechanism.

The Programming Model

- 8086 through Core2 considered **program visible.**
  - registers are used during programming and are specified by the instructions
- Other registers considered to be **program invisible.**
  - not addressable directly during applications programming

- 80286 and above contain program-invisible registers to control and operate protected memory.
  - and other features of the microprocessor
- 80386 through Core2 microprocessors contain full 32-bit internal architectures.
- 8086 through the 80286 are fully upward-compatible to the 80386 through Core2.
- Figure 2–1 illustrates the programming model 8086 through Core2 microprocessor.
  - including the 64-bit extensions

Figure 2–1 The programming model of the 8086 through the Core2 microprocessor including the 64-bit extensions.



Multipurpose Registers

- **RAX -** a 64-bit register (RAX), a 32-bit register **(accumulator)** (EAX), a 16-bit register (AX), or as either of two 8-bit registers (AH and AL).

- The accumulator is used for instructions such as multiplication, division, and some of the adjustment instructions.

- Intel plans to expand the address bus to 52 bits to address 4P (2^{52}~10^{15} =peta) bytes of memory.

Address Space (Main Memory)

- Address bus:16 bit →Address Space:64 KBytes
- Address bus:20 bit →Address Space:1 MBytes
  Address bus:32 bit →Address Space:4 GBytes
  Address bus:34 bit →Address Space:16GBytes
- Address bus:36 bit →Address Space:64GBytes
- Address bus:38 bit →Address Space:256GBytes
- Address bus:52 bit →Address Space:10^{15} Bytes

- **RBX,** addressable as RBX, EBX, BX, BH, BL.
  - BX register **(base index)** sometimes holds offset address of a location in the memory system in all versions of the microprocessor
- **RCX**, as RCX, ECX, CX, CH, or CL.
  - a **(count)** general-purpose register that also holds the count for various instructions
- **RDX,** as RDX, EDX, DX, DH, or DL.
  - a **(data)** general-purpose register
  - holds a part of the result from a multiplication or part of dividend before a division

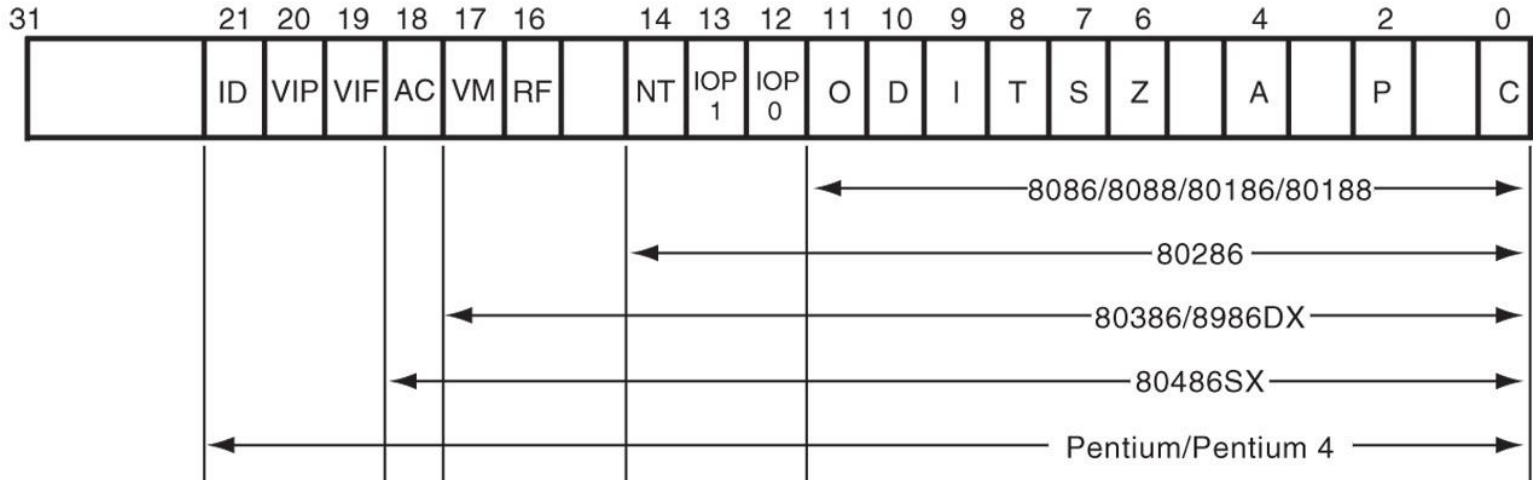
- **RBP,** as RBP, EBP, or BP.
  - points to a memory **(base pointer)** location for memory data transfers
- **RDI** addressable as RDI, EDI, or DI.
  - often addresses **(destination index)** string destination data for the string instructions
- **RSI** used as RSI, ESI, or SI.
  - the **(source index)** register addresses source string data for the string instructions
  - like RDI, RSI also functions as a general-purpose register

Special-Purpose Registers

- Include RIP, RSP, and RFLAGS
  - segment registers include CS, DS, ES, SS, FS, and GS
- **RIP** addresses the next instruction in a section of memory.
  - defined as **(instruction pointer)** a code segment
- **RSP** addresses an area of memory called the stack.
  - the **(stack pointer)** stores data through this pointer

- **RFLAGS** indicate the condition of the microprocessor and control its operation.
- Figure 2–2 shows the flag registers of all versions of the microprocessor.
- Flags are upward-compatible from the 8086/8088 through Core2 .
- The rightmost five and the overflow flag are changed by most arithmetic and logic operations.
  - although data transfers do not affect them

Figure 2–2 The EFLAG and FLAG register counts for the entire 8086 and Pentium microprocessor family.



- Flags never change for any data transfer or program control operation.
- Some of the flags are also used to control features found in the microprocessor.

2–2 REAL MODE MEMORY ADDRESSING

- 80286 and above operate in either the real or protected mode.
- **Real mode operation** allows addressing of only the first 1M byte of memory space—even in Pentium 4 or Core2 microprocessor.
  - the first 1M byte of memory is called the **real memory**, **conventional memory**, or **DOS memory** system

Segments and Offsets

- All real mode memory addresses must consist of a segment address plus an offset address.
  - **segment address** defines the beginning address of any 64K-byte memory segment
  - **offset address** selects any location within the 64K byte memory segment
- Figure 2–3 shows how the **segment plus offset** addressing scheme selects a memory location.

History

- Late 70's: 8086, Real Mode and has no protection.
- 1982: 80286, Real Mode and 16b Protected Mode.
- 1985: 80386, Real Mode and 32b Protected Mode.

2–3 INTRO TO PROTECTED MODE MEMORY ADDRESSING

- Allows access to data and programs located within & above the first 1M byte of memory.
- **Protected mode** is where Windows operates.
- In place of a segment address, the segment register contains a **selector** that selects a descriptor from a descriptor table.
- The **descriptor** describes the memory segment's location, length, and access rights.

Selectors and Descriptors

- The **selector** is located in the segment register
  - it selects one of 8192 descriptors from one of two tables of descriptors
  - The descriptor describes the location, length, and access rights of the segment of memory.
- Indirectly, the register still selects a memory segment, but not directly as in real mode.

- **Global descriptors** contain segment definitions that apply to all programs.
- **Local descriptors** are usually unique to an application.
  - a global descriptor might be called a **system descriptor**, and local descriptor an **application descriptor**
- Figure 2–6 shows the format of a descriptor for the 80286 through the Core2.
  - each descriptor is 8 bytes in length
  - global and local descriptor tables are a maximum of 64K bytes in length

PROTECTED MODE MEMORY ADDRESSING

- In protected addressing mode segments can be of variable size(below or above 64 KB).

- The offset part of the memory address is still the same as in real addressing mode. However, when in the protected mode, the processor can work either with 16-bit offsets or with 32-bit offsets.

PROTECTED MODE MEMORY ADDRESSING

- A 32-bit offset allows segments of up to 4G bytes. Notice that in real-mode the only available instruction mode is the 16-bit mode.

- One difference between real and protected addressing mode is that the segment address, as discussed with real mode memory addressing, is no longer present in the protected mode.

PROTECTED MODE MEMORY ADDRESSING

- In place of the segment address, the segment register contains a selector
- The selector selects a descriptor from a descriptor table.
- The descriptor describes the memory segment's location, length, and access rights. This is similar to selecting one card from a deck of cards in one's pocket.

PROTECTED MODE MEMORY ADDRESSING

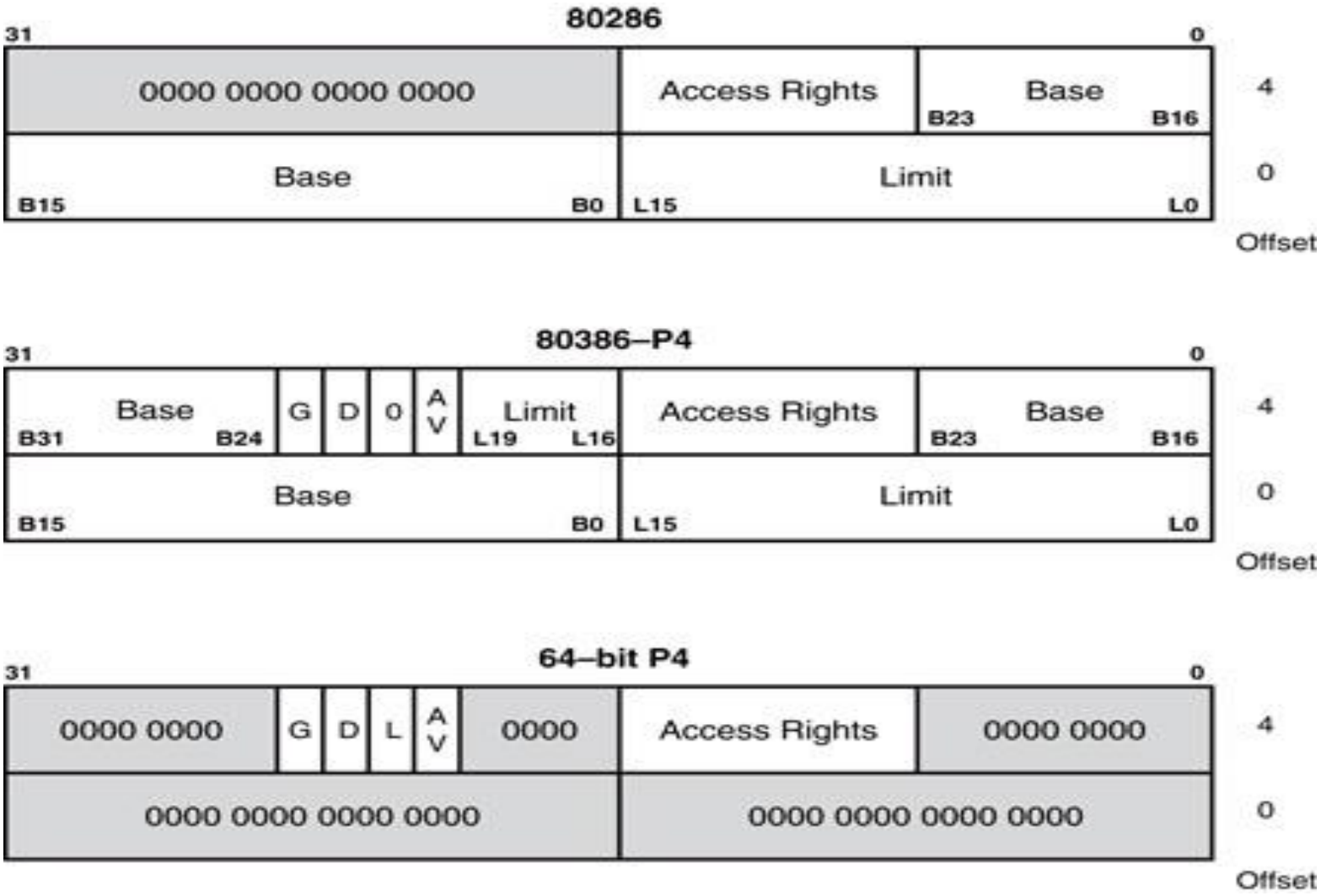
- The selector, located in the segment register, selects one of 8192 descriptors from one of two tables of descriptors (stored in memory): the global and local descriptor tables. The descriptor describes the location, length and access rights of the memory segment.

- Each descriptor is 8 bytes long.

PROTECTED MODE MEMORY ADDRESSING

- The 8192 descriptor table requires 8 * 8192 = 64K bytes of memory.

Figure 2–6 The 80286 through Core2 64-bit descriptors.

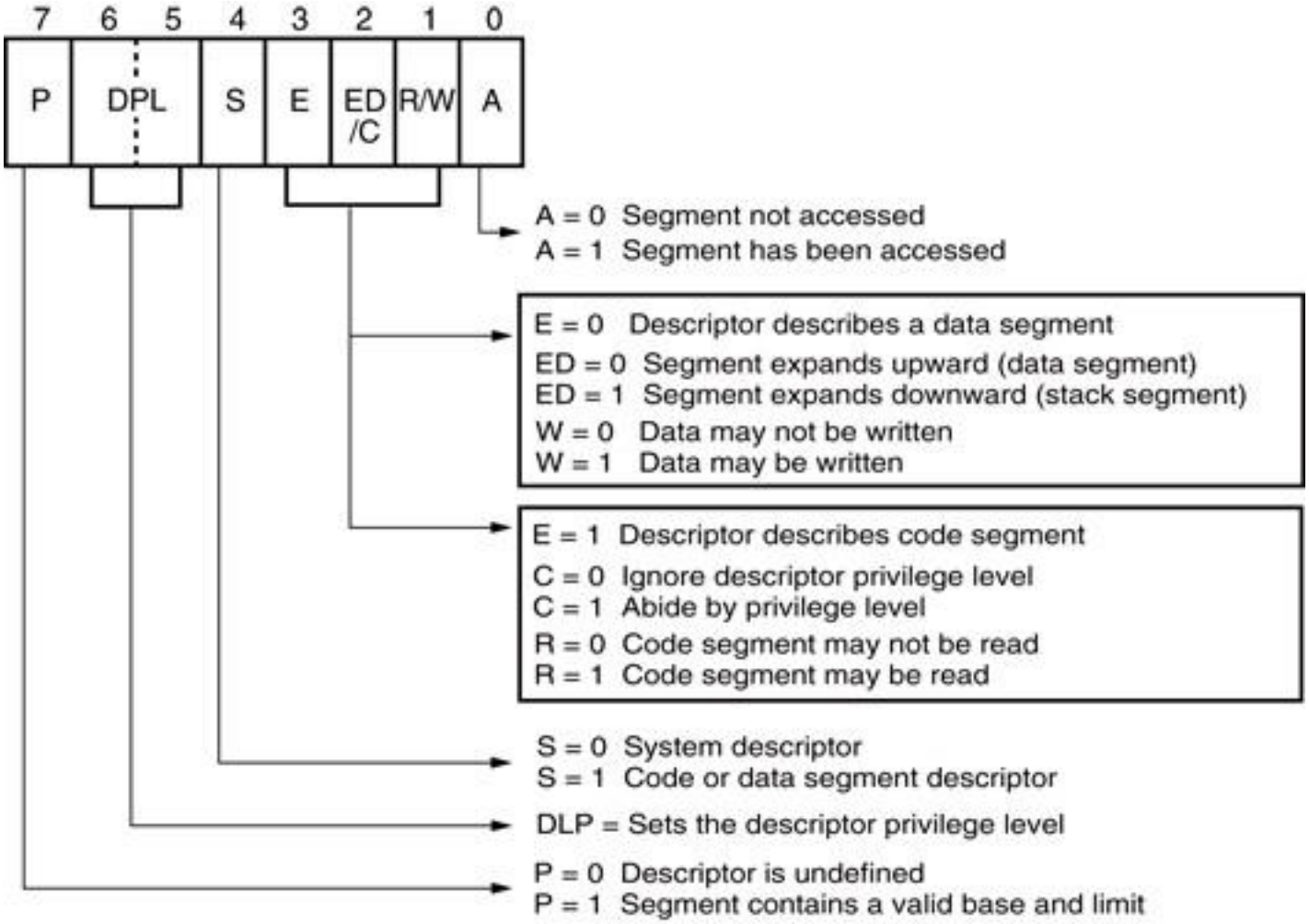


- The **base address** of the descriptor indicates the starting location of the memory segment.
- The segment length can be 4K to 4G bytes in steps of 4K bytes.
  - 32-bit offset address allows segment lengths of 4G bytes
  - 16-bit offset address allows segment lengths of 64K bytes.

- Operating systems operate in a 16- or 32-bit environment.
- DOS uses a 16-bit environment.
- Most Windows applications use a 32-bit environment called **WIN32**.

- The **access rights byte** controls access to the protected mode segment.
  - describes segment function in the system and allows complete control over the segment
  - if the segment is a data segment, the direction of growth is specified
- You can specify whether a data segment can be written or is **write-protected**.

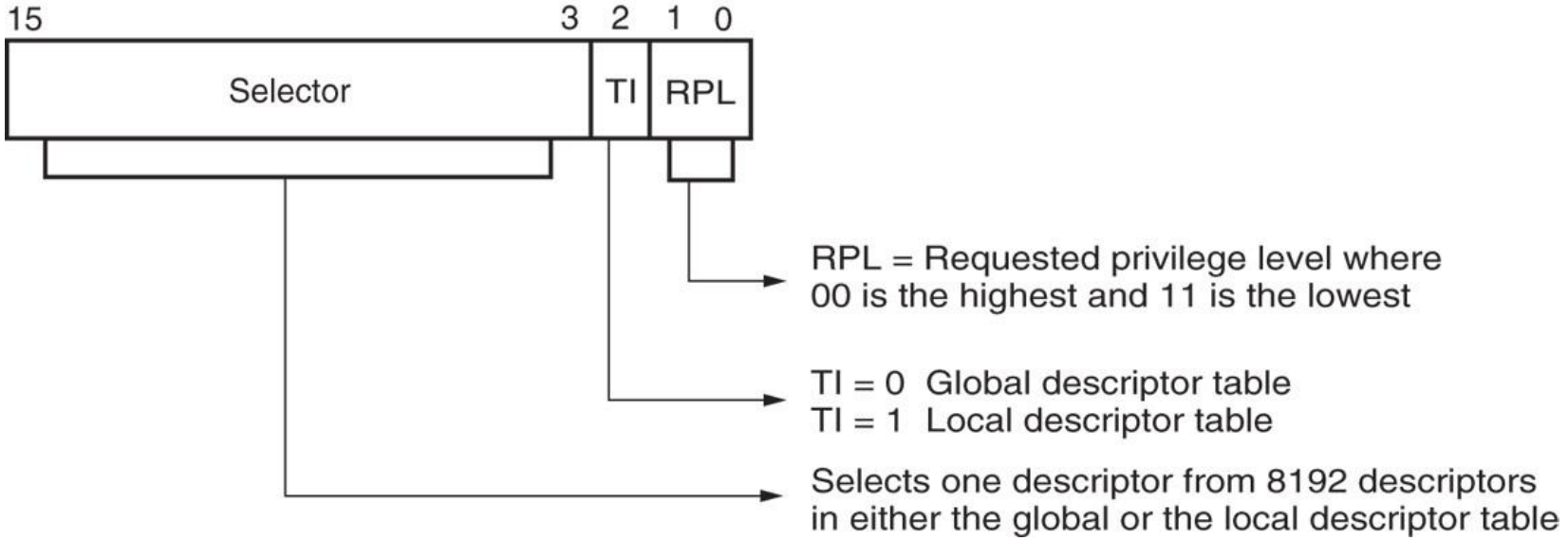
Figure 2–7 The access rights byte for the 80286 through Core2 descriptor.



Note: Some of the letters used to describe the bits in the access rights bytes vary in Intel documentation.

- Descriptors are chosen from the descriptor table by the selector in the segment register.
  - register contains a 13-bit selector field, a table selector bit, and requested privilege level field
- The **TI bit** selects either the global or the local descriptor table.
- **Requested Privilege Level** (RPL) requests the access privilege level of a memory segment.

Figure 2–8 The contents of a segment register during protected mode operation of the 80286 through Core2 microprocessors.



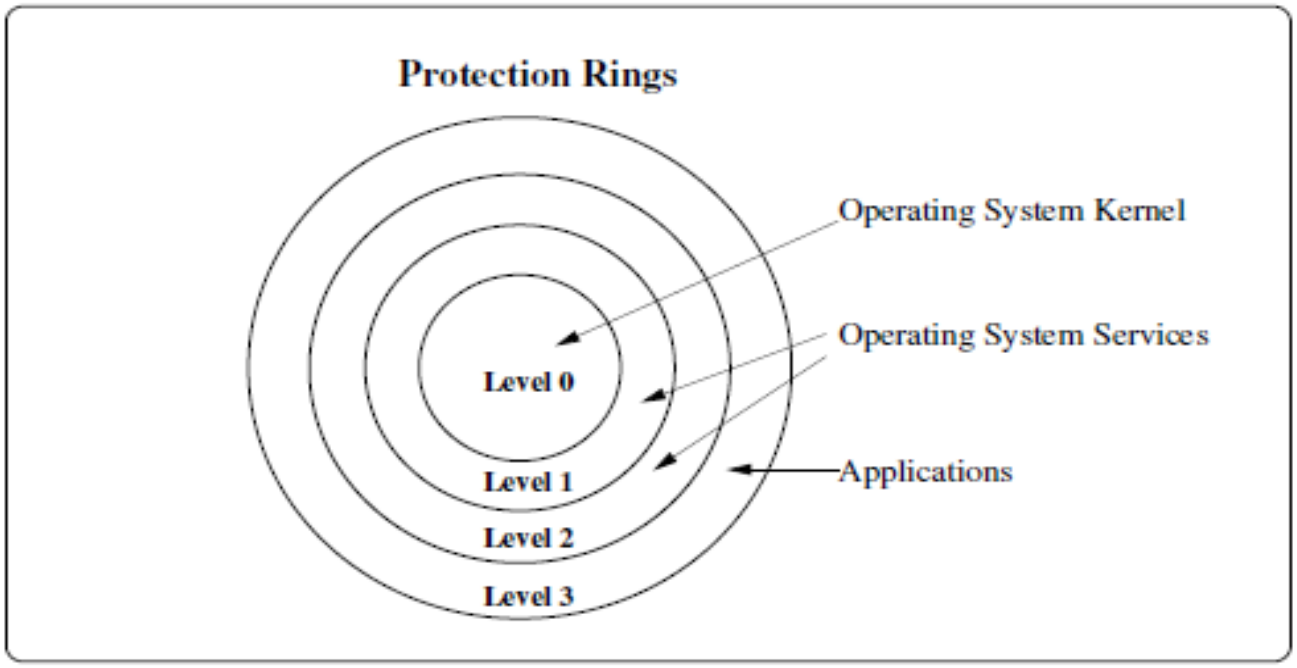
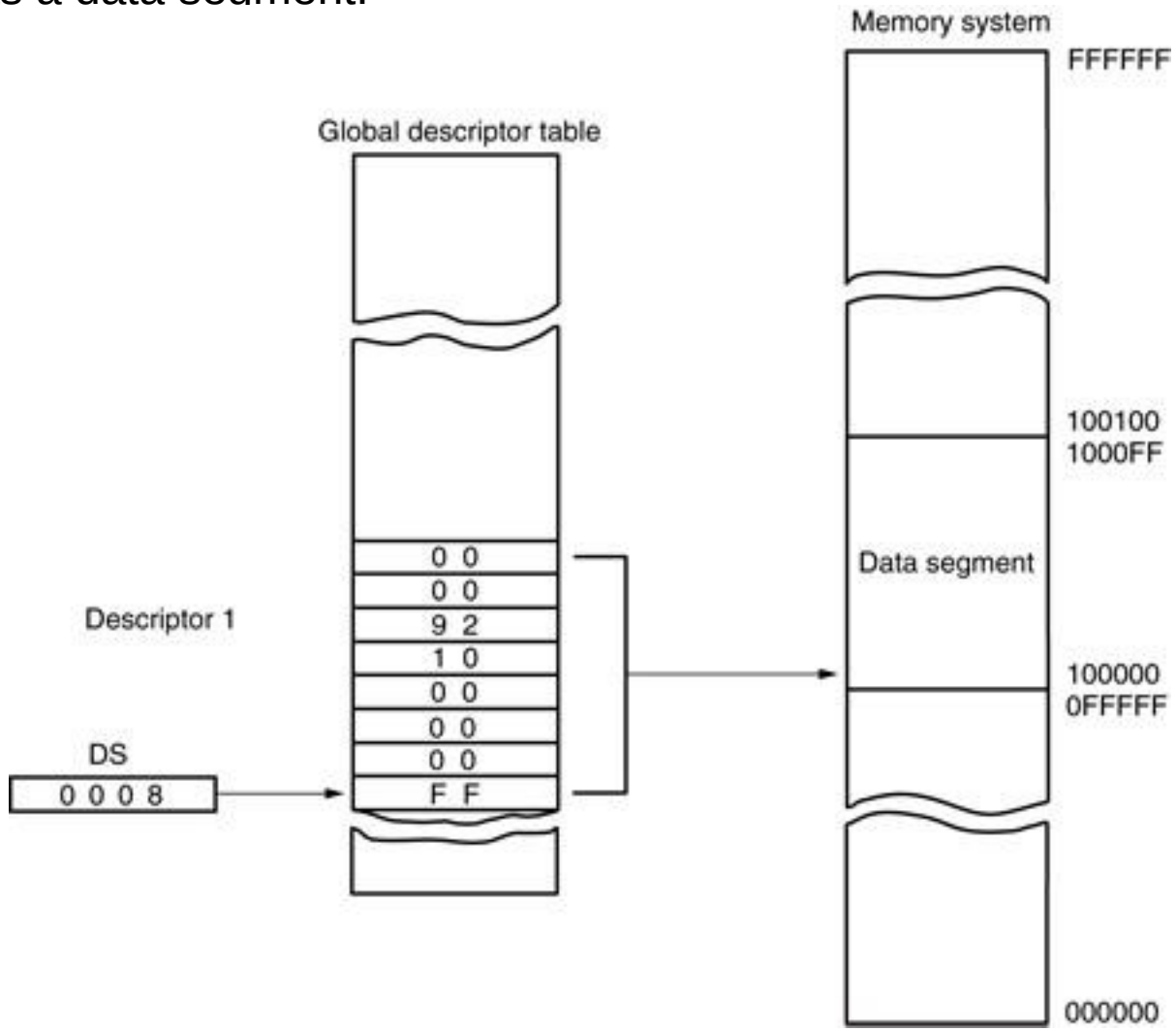


Figure: Rings

Memory protection across ring boundaries: once we divide the memory into several rings, we can
define security policies based on rings. For example, we can prevent code in ring 3 from accessing
data in ring 0, etc.

- Figure 2–9 shows how the segment register, containing a selector, chooses a descriptor from the global descriptor table.
- The entry in the global descriptor table selects a segment in the memory system.
- Descriptor zero is called the null descriptor, must contain all zeros, and may not be used for accessing memory.

Figure 2–9 Using the DS register to select a description from the global descriptor table. In this example, the DS register accesses memory locations 00100000H–001000FFH as a data segment.



Memory Addressing
Address Translation Mechanism of 80386

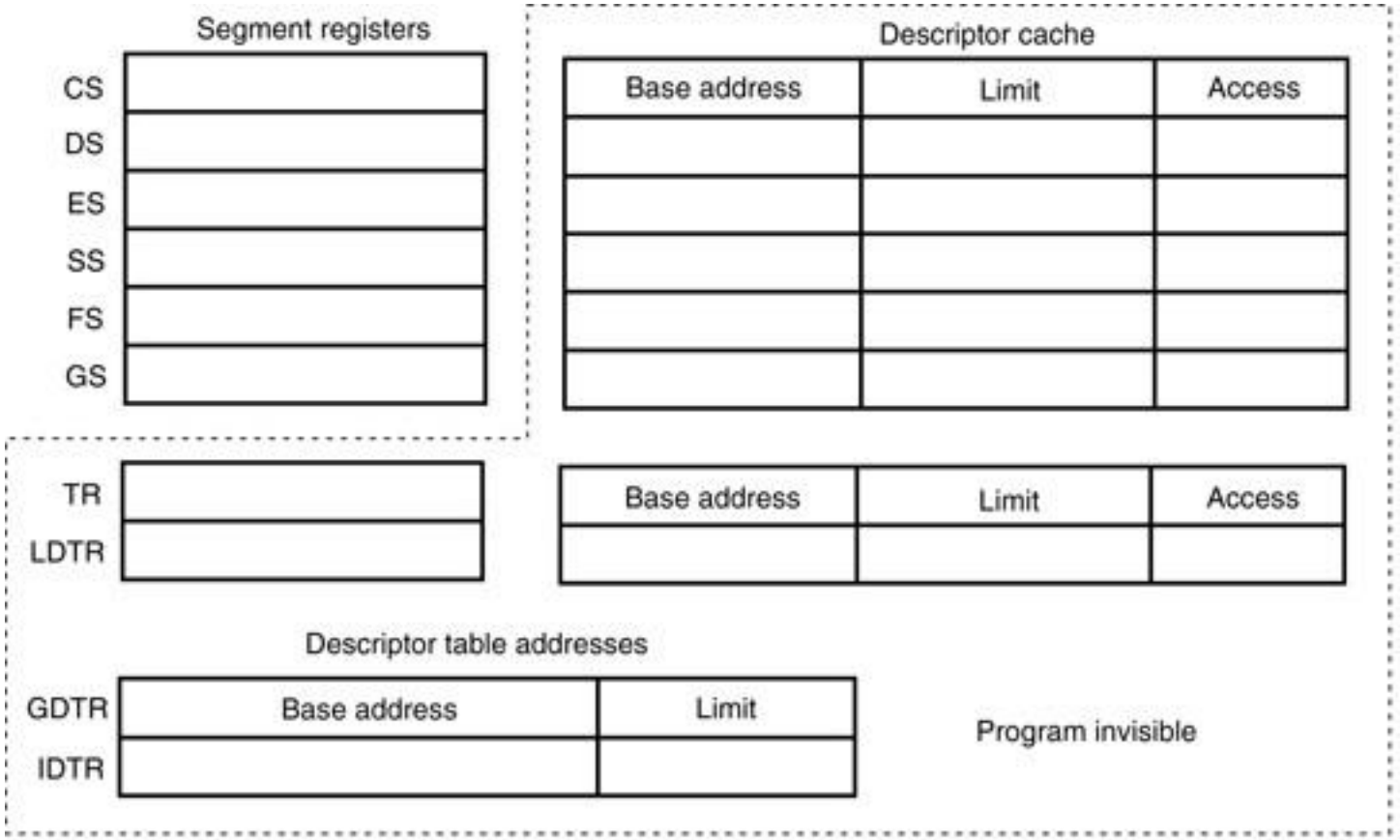
Protected Mode Addressing Mechanism

- 80386 transforms logical addresses into physical address two steps:
- **Segment translation:** a logical address is converted to a linear address.
- **Page translation:** a linear address is converted to a physical address.(optional)
- These translations are performed in a way that is not visible to applications programmers.

Program-Invisible Registers

- Global and local descriptor tables are found in the memory system.
- To access & specify the table addresses, 80286–Core2 contain program-invisible registers.
  - not directly addressed by software
- Each segment register contains a program-invisible portion used in the protected mode.

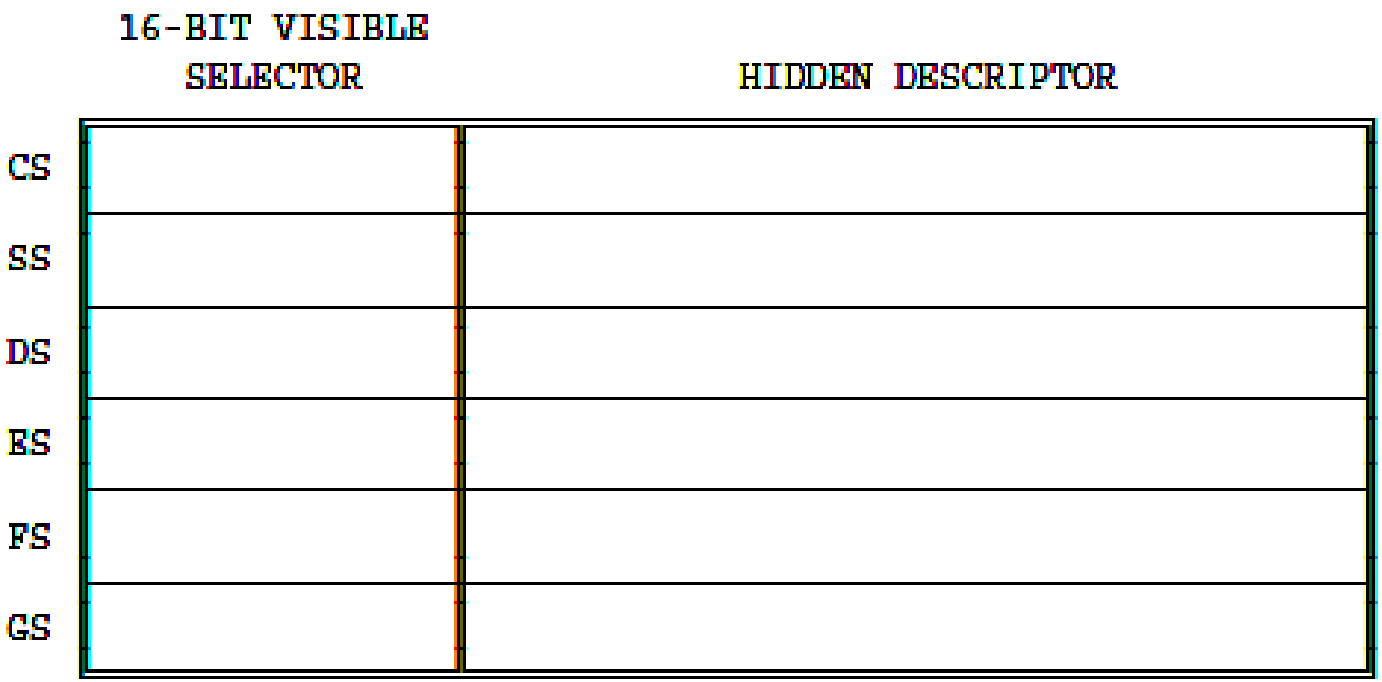
Figure 2–10 The program-invisible register within the 80286–Core2 microprocessors.



Notes:

1. The 80286 does not contain FS and GS nor the program-invisible portions of these registers.
2. The 80286 contains a base address that is 24-bits and a limit that is 16-bits.
3. The 80386/80486/Pentium/Pentium Pro contain a base address that is 32-bits and a limit that is 20-bits.
4. The access rights are 8-bits in the 80286 and 12-bits in the 80386/80486/Pentium–Core2.

Figure 5-7. Segment Registers



Memory Paging

- In the paging memory-management scheme, the operating system retrieves data from secondary storage in same-size blocks called pages.

- The Memory management unit consists of
  - Segmentation unit and
  - Paging unit.
- Segmentation unit allows segments of size 4Gbytes at max.

- The Paging unit organizes the physical memory in terms of pages of 4kbytes size each.

Protected Mode Addressing Mechanism

- 80386 transforms logical addresses into physical address two steps:
- **Segment translation:** a logical address is converted to a linear address.
- **Page translation:** a linear address is converted to a physical address.(optional)
- These translations are performed in a way that is not visible to applications programmers.

- Mechanism available in the 80386 and up.

- Allows a linear address (program generated) of a program to be located in any portion of physical memory.

- Paging unit works under the control of the segmentation unit, i.e. each segment is further divided into pages.

- The linear address is invisibly translated to any physical address

- Through paging to the hard disk drive and paging to the memory through the memory paging unit, any Windows application can be executed

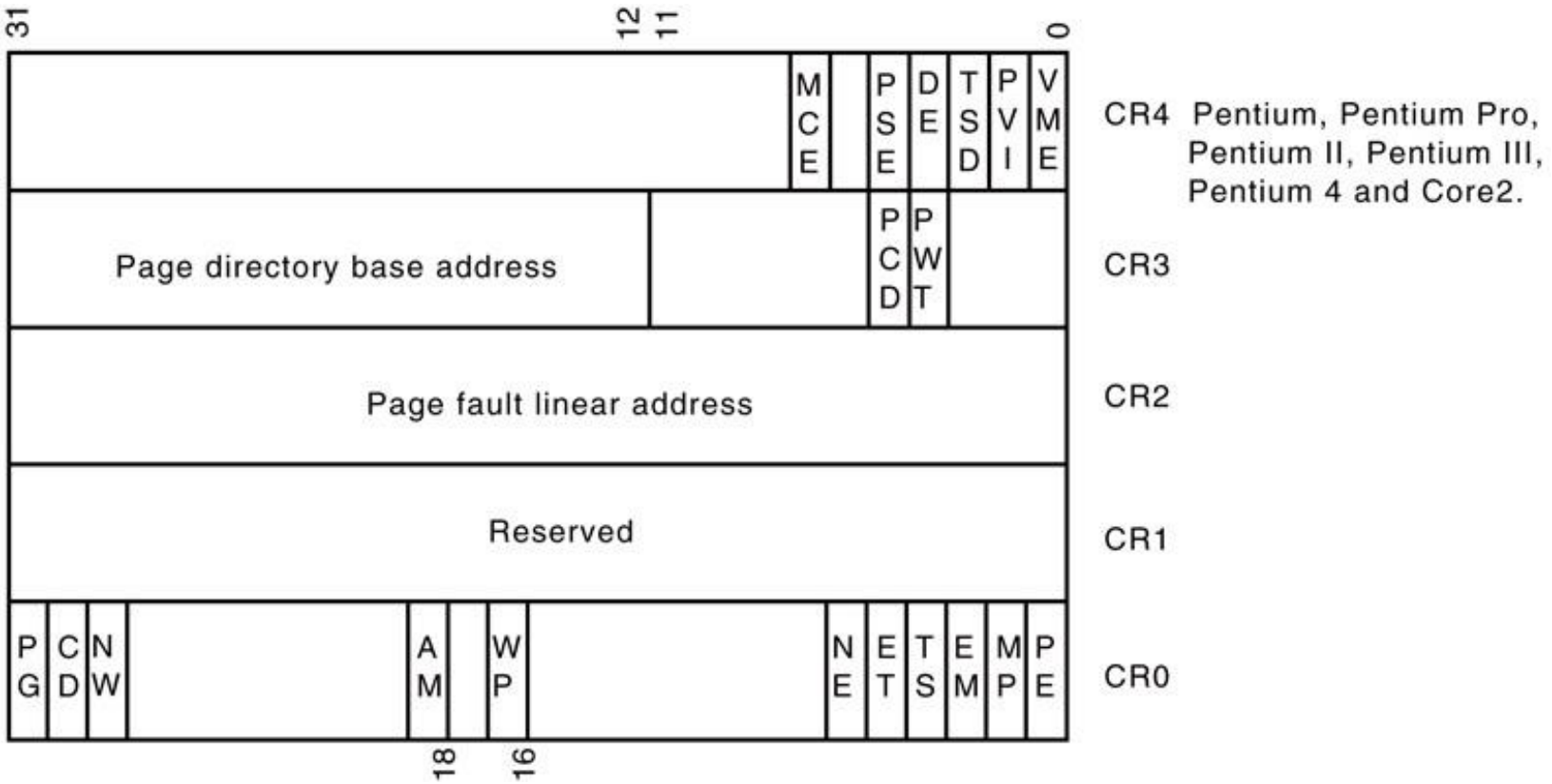
MEMORY PAGING

- The **memory paging mechanism** allows any physical memory location to be assigned to any linear address.
- **linear address** is defined as the address generated by a program.
- **Physical address** is the actual memory location accessed by a program.
- With memory paging, the linear address is invisibly translated to any physical address.

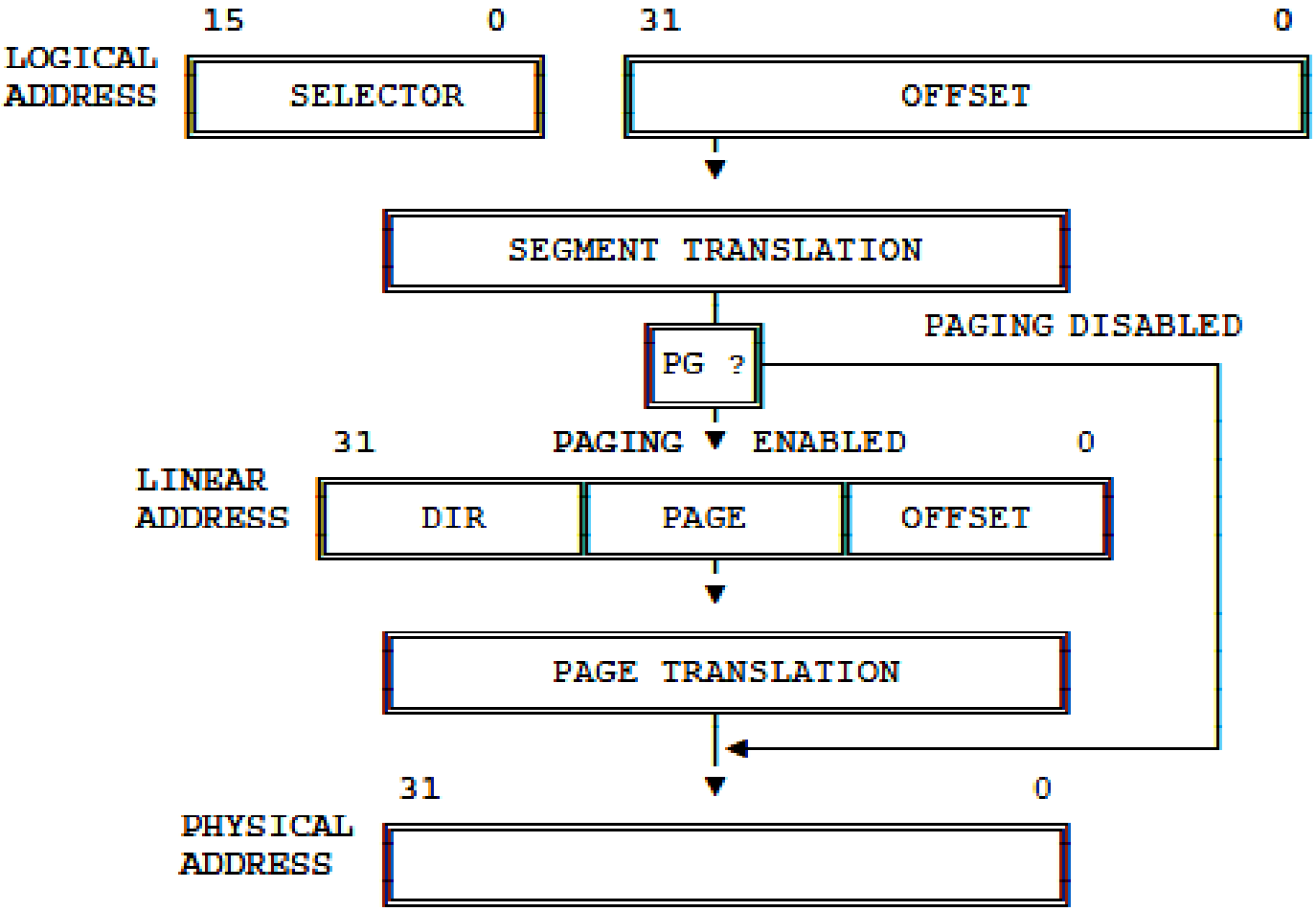
Paging Registers

- The paging unit is controlled by the contents of the microprocessor's control registers.
- Beginning with Pentium, an additional control register labeled CR4 controls extensions to the basic architecture.
- See Figure 2–11 for the contents of control registers CR0 through CR4.

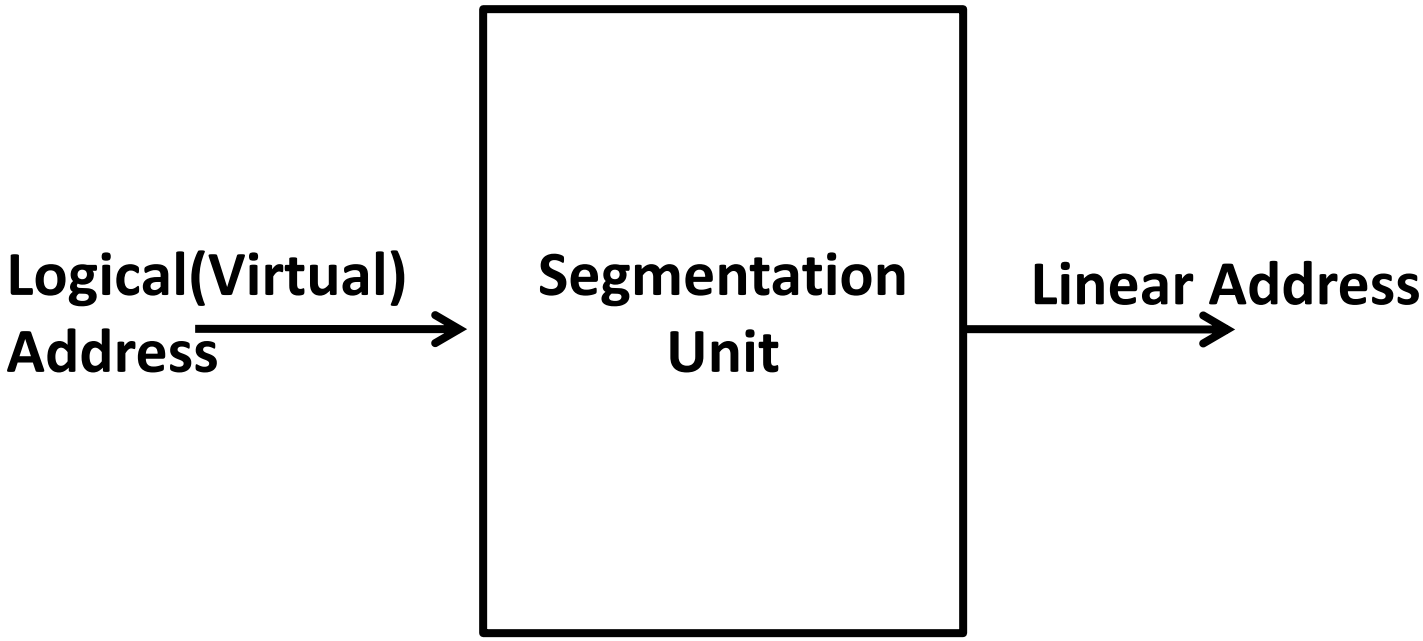
Figure 2–11 The control register structure of the microprocessor.

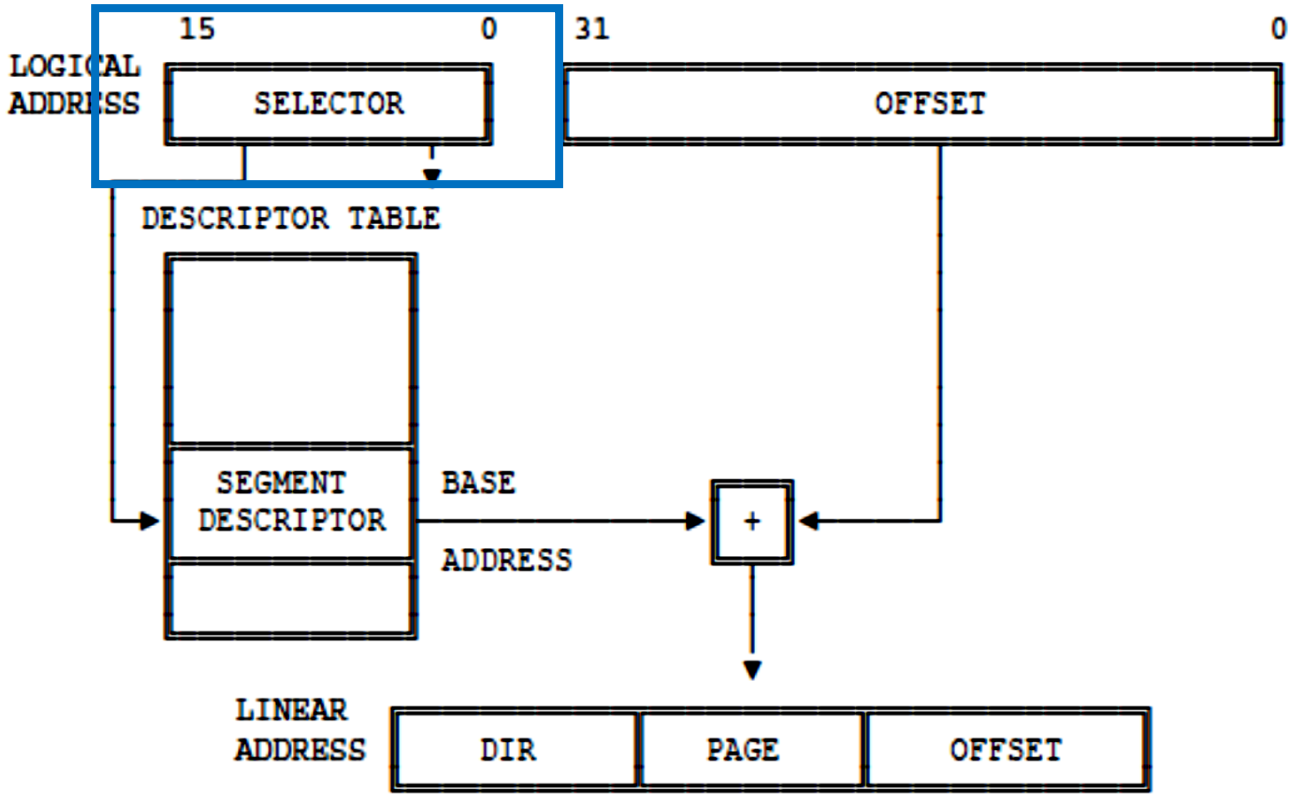


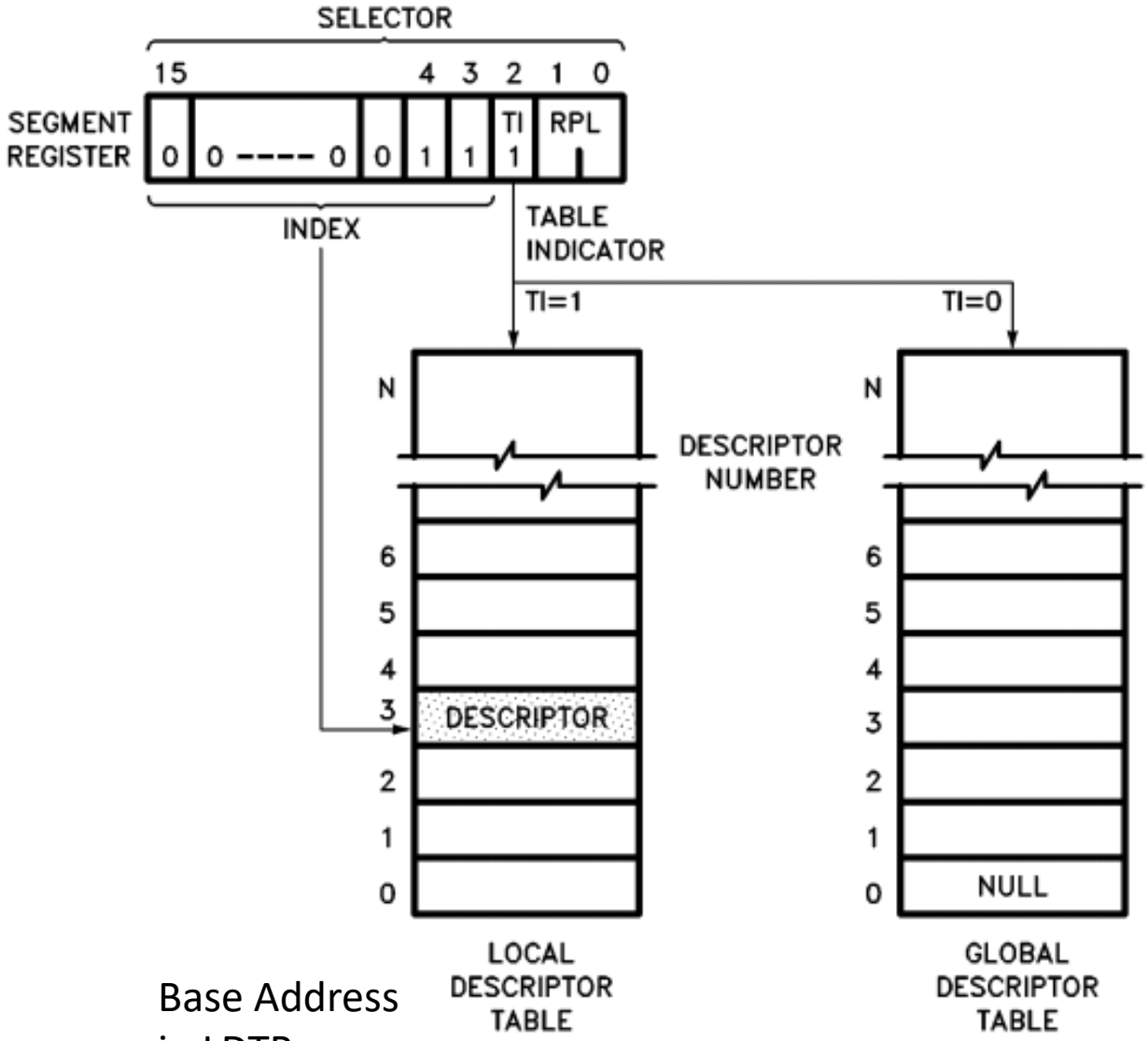
- The following figure illustrates the two translations:



Segmentation

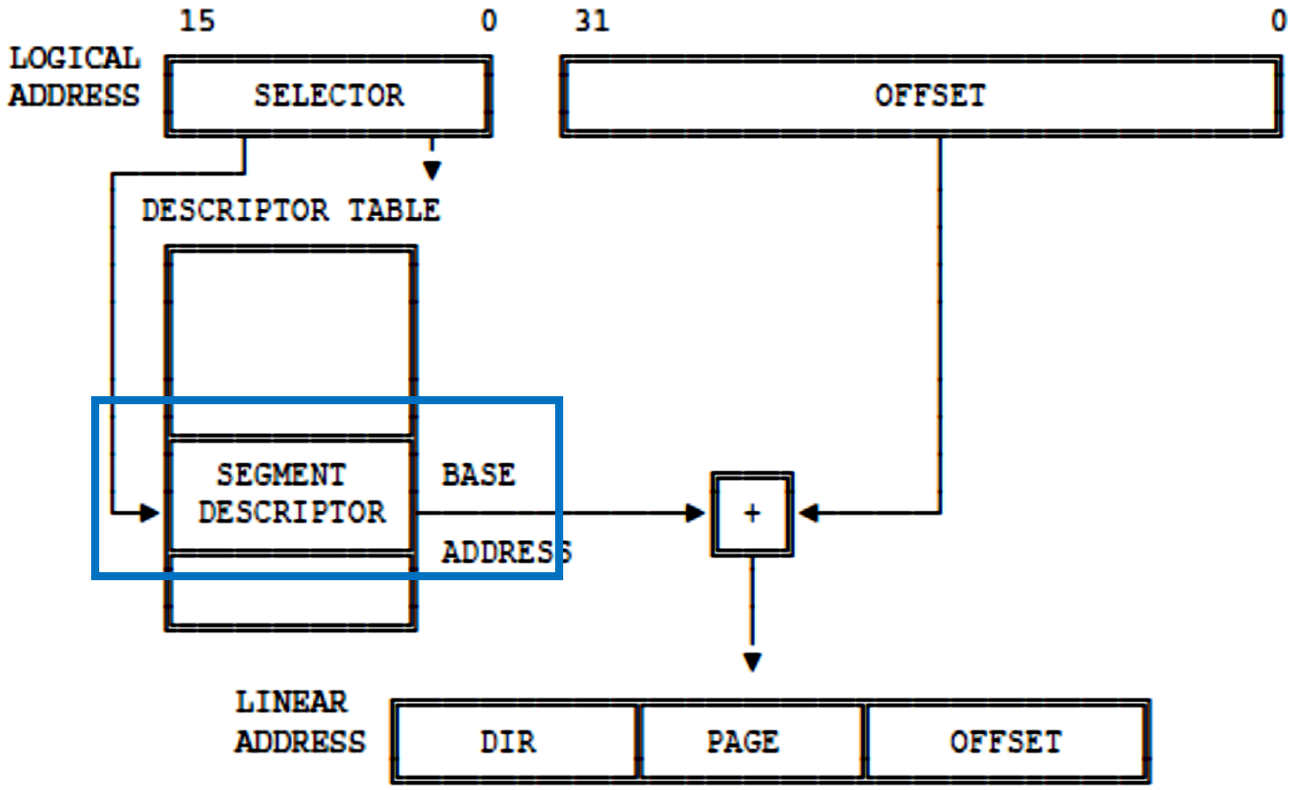

Logical(Virtual)
Address
Segmentation
Unit
Linear Address


15
0
31
0
LOGICAL ADDRESS
SELECTOR
OFFSET
DESCRIPTOR TABLE
SEGMENT DESCRIPTOR
BASE
ADDRESS
+
LINEAR ADDRESS
DIR
PAGE
OFFSET

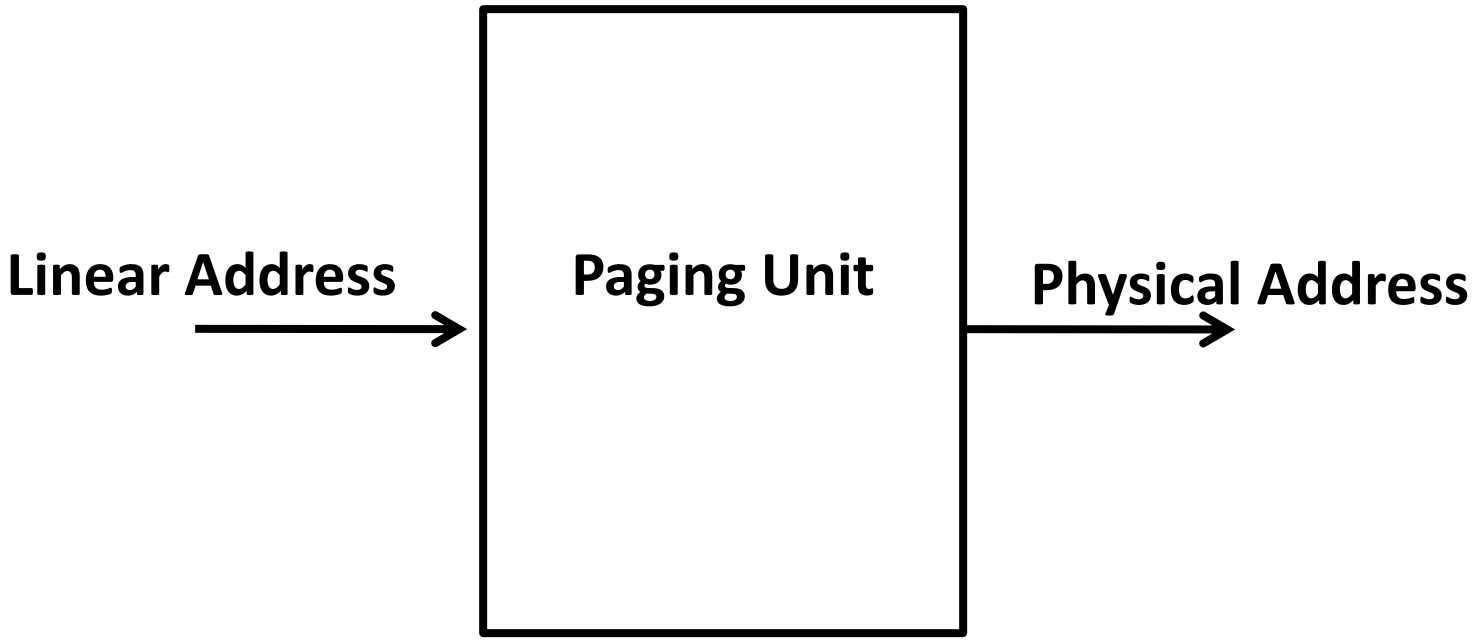


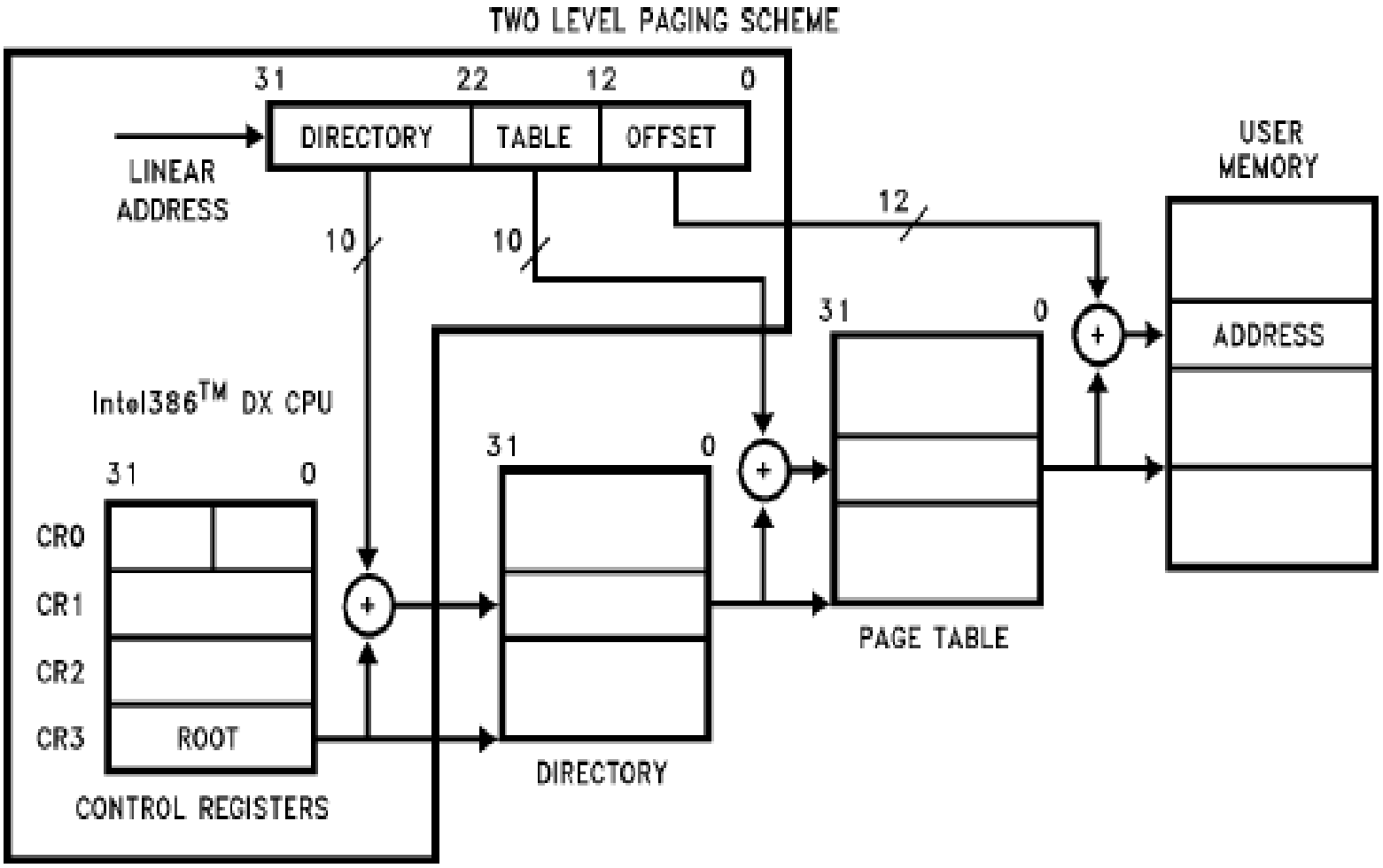
Base Address in LDTR Register

Base Address in GDTR Register


15
0
31
0
LOGICAL
ADDRESS
SELECTOR
OFFSET
DESCRIPTOR TABLE
SEGMENT
DESCRIPTOR
BASE
ADDRESS
+
LINEAR
ADDRESS
DIR
PAGE
OFFSET

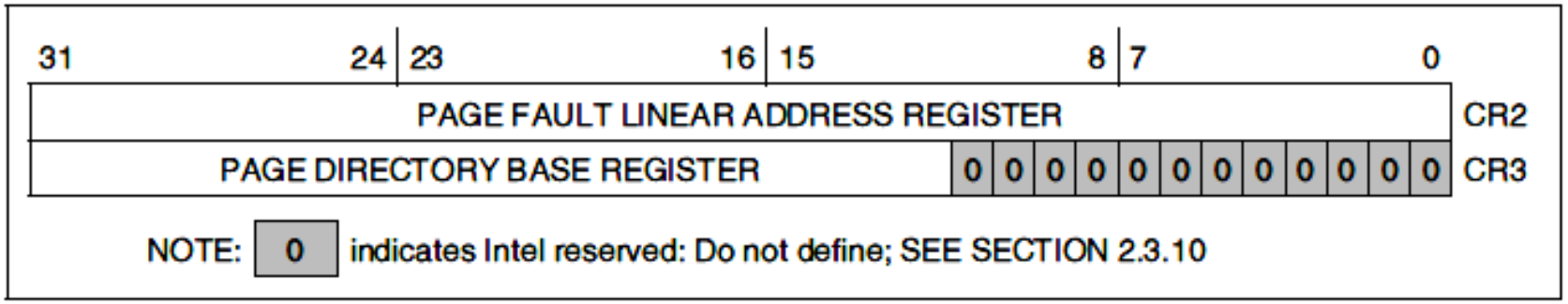
Paging


Linear Address
Paging Unit
Physical Address


TWO LEVEL PAGING SCHEME
31
22
12
0
DIRECTORY
TABLE
OFFSET
LINEAR ADDRESS
10
10
12
USER MEMORY
31
0
ADDRESS
Intel386™ DX CPU
31
0
31
0
CR0
CR1
CR2
CR3
ROOT
PAGE TABLE
DIRECTORY
CONTROL REGISTERS

Page Descriptor Base Register

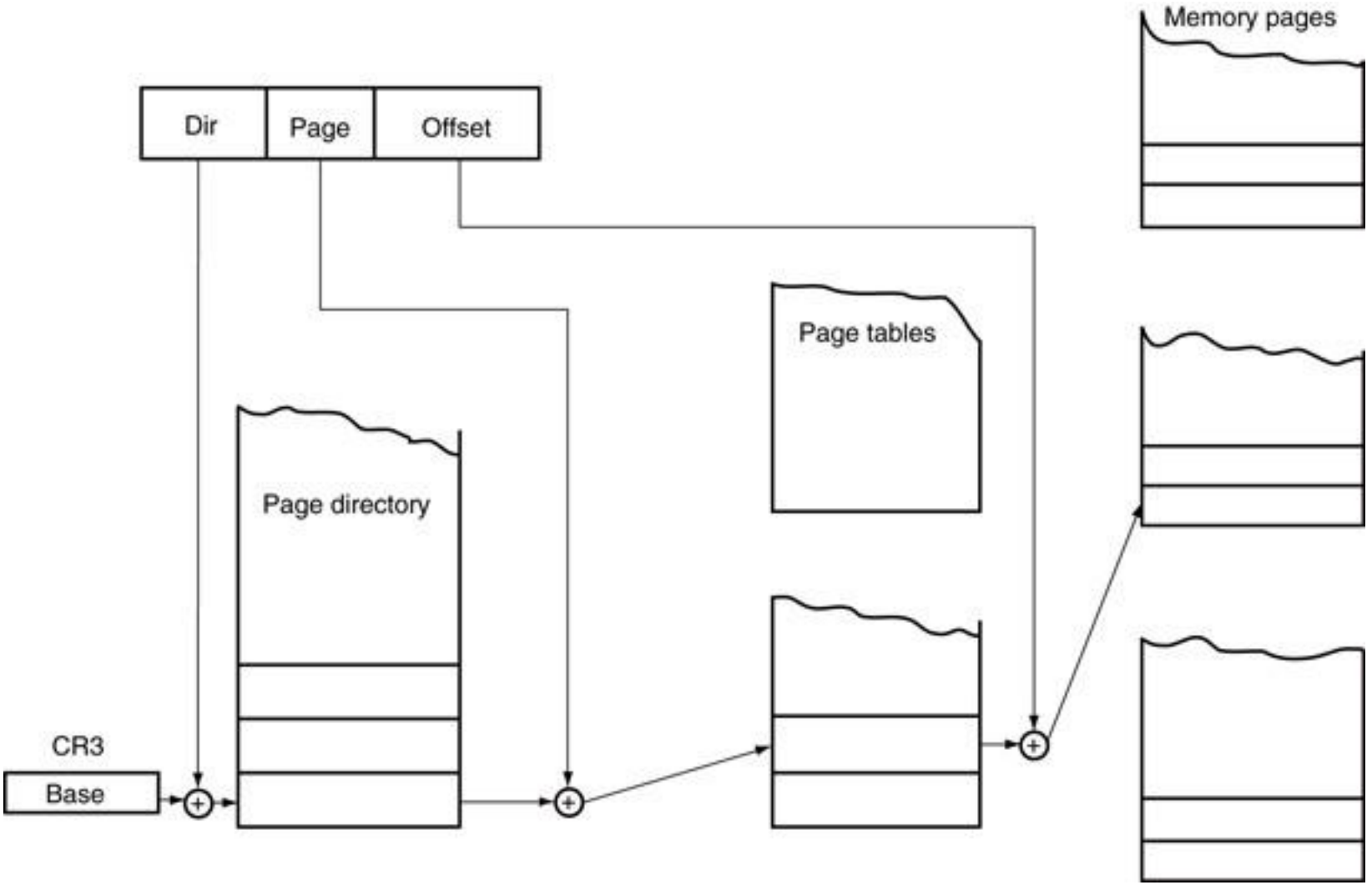
- CR2 is used to store the 32-bit linear address of page fault.
- CR3 (Page Directory Physical Base Address Register) stores the **physical starting address** of Page Directory.



Page Directory

- It is at the most **4KB in size** and allows upto **1024** entries are allowed.
- The **upper 10 bits** of the linear address are used as an **index** to corresponding page directory entry
- Page directory entry **points** to page tables.
- Only one page directory in the system.

Figure 2–13 The paging mechanism in the 80386 through Core2 microprocessors.



- Each entry in the page directory corresponds to 4M bytes of physical memory.
- Each entry in the page table repages 4K bytes of physical memory.
- Windows also repages the memory system.

Count….

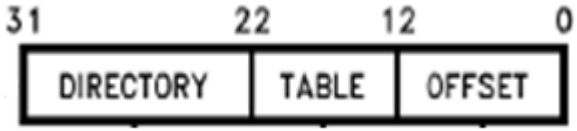
The virtual address is broken into three pieces

- ***Directory*** *: Each* **page directory** *addresses a 4MB section of main mem.*
- ***Page Table*** *: Each* **page table** *entry addresses a 4KB section of main mem.*
- ***Offset*** *: Specifies the byte in the page.*

Example

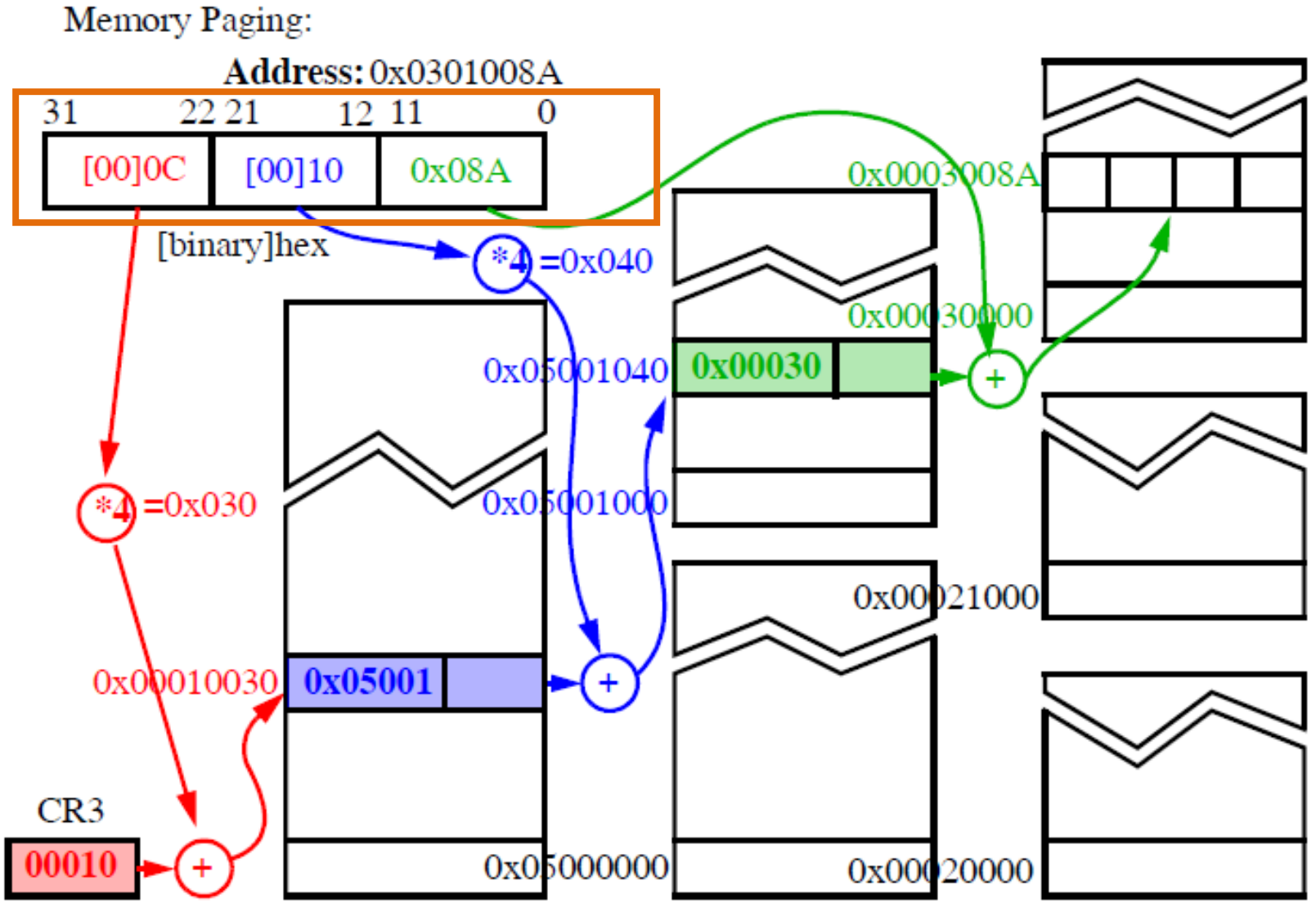
Linear Address : 0301008A

0000 0011 0000 0001 0000 0000 1000 1010



Binary	**00 0000 1100 (10bits)**	**00 0001 0000 (10bits)**	**0000 1000 1010 (12bits)**
Hex	**00C**	**010**	**08A**

Example

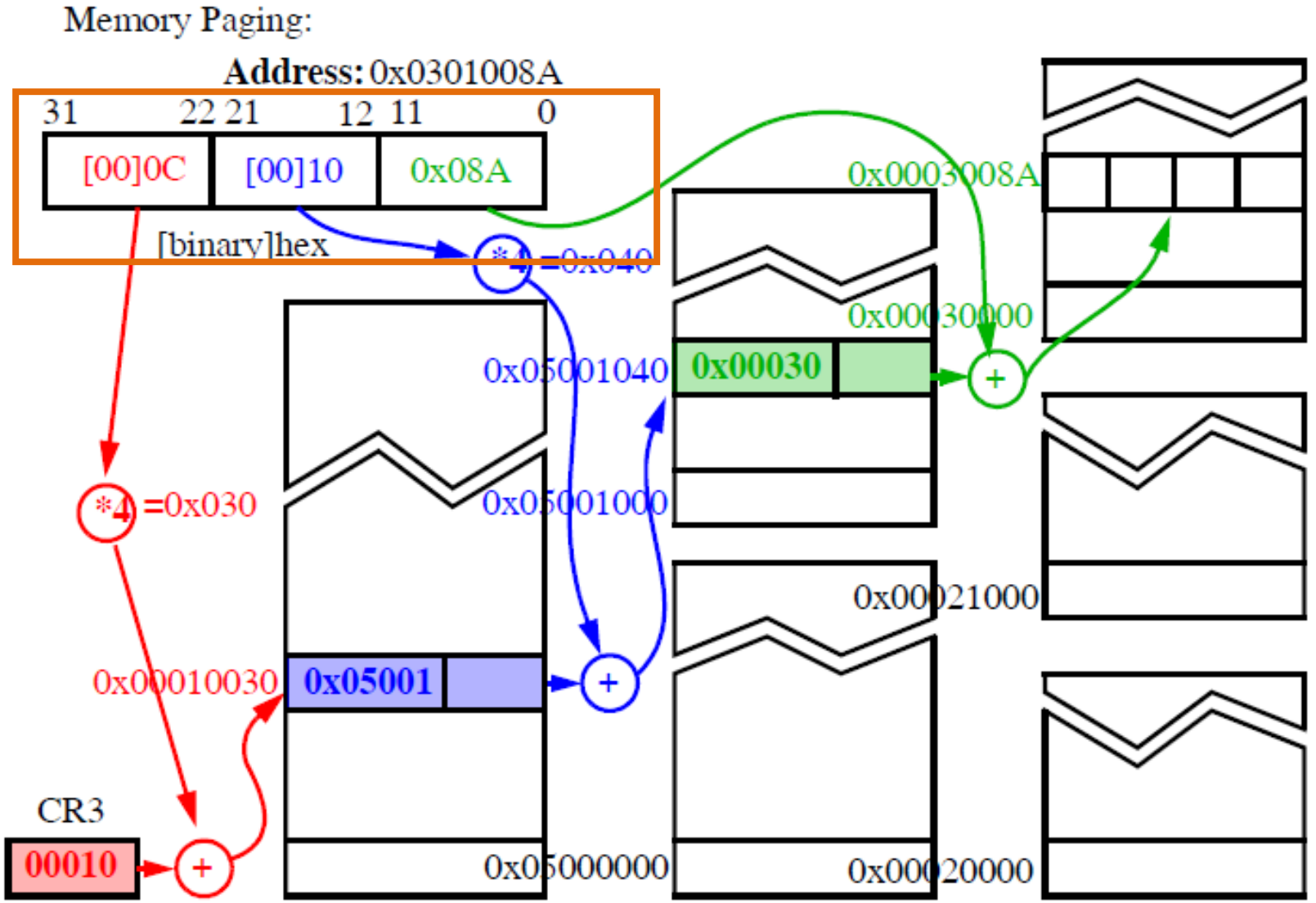

Memory Paging:
Address: 0x0301008A
31
22 21
12 11
0
[00]0C
[00]10
0x08A
[binary]hex
*4 =0x040
*4 =0x030
0x0003008A
0x00030000
0x05001040
0x00030
0x05001000
0x00021000
0x00010030
0x05001
CR3
00010
0x05000000
0x00020000

Hex	00C(DIR)	x4	030
Binary	00 0000 1100	00 0000 1100 x 0100 _____ 00 0011 0000	

CR3	+	DIR*4	=	Index to PDE
(20-bit)		(12-bit)		

00010H	+	030H	=	00010030H

Example


Memory Paging:
Address: 0x0301008A
31
22 21
12 11
0
[00]0C
[00]10
0x08A
[binary]hex
*4 =0x040
*4 =0x030
0x0003008A
0x00030000
0x05001040
0x00030
0x05001000
0x00021000
0x00010030
0x05001
CR3
00010
0x05000000
0x00020000